AlgoTch: the Virtual Teacher within AlgoPath

E. Perrin [*], S. Linck [**], D. Zekrouf [**]

** HENALLUX, Marche-en-Famenne, Belgium*

CReSTIC, Reims Champagne-Ardenne University, France Email: estelle.perrin@univ-reims.fr

*** CReSTIC, IFTS*

Reims Champagne-Ardenne University, Charleville-Mézières, France

Email: sebastien.linck@univ-reims.fr, dimitry.zekrouf@etudiant.univ-reims.fr

Abstract. *AlgoPath is a serious game to help students with no classical computer science background understand the main concepts of algorithmics. The player builds a world that represents the ongoing algorithm by herself/himself. This world is made of concrete, paths and grass. Thanks to the inner model of AlgoPath that is based on the Model-View-Controller architecture (MVC), we intend to create a virtual teacher in the world of AlgoPath named AlgoTch. The main goals of AlgoTch are to oversee the players' interactions, to intervene when the interactions do not correspond to what it is expected from her/him and to give hints and tips when she/he asks a question. We show how AlgoTch interacts with the Controller component of the architecture and we describe the inner model of AlgoTch. The latter is based on particular comments the real teacher writes for the various mistakes that can be made by a player and on first degree logic. We explain the three gameplay types available within AlgoPath: a single one, in which the gamer can discover the design of algorithms by her/his own; a competitive one, in which two or several players fight against one another to be the first to find the best algorithm; and a collaborative one, in which they have to find the best algorithm as a team. Finally, we explain our choice to develop AlgoTch in a new version of AlgoPath with Unity.*

Keywords: *virtual environment, learning, virtual teacher, serious game, algorithmics*

1. Introduction

Today's students are not anymore the people they used to be. Their lives depend on a world full of electronics. Smart phones, tablets and computers (preferably connected to the Internet) provide them with video games, chats, e-books, information at any time. These students, born after the eighties, are called digital natives as opposed to digital immigrants [Pre01]. They live surrounded by multimedia but sometimes - strangely - multimedia seem to stay away from their teaching space. That is particularly true with algorithmics courses where the teacher asks them to work with a pen and a sheet of paper. Algorithmics courses are the basics of computer science but computers remain away from them. Digital natives might find this particularly odd and consequences could be dramatic because more and more people use electronics and less and less are eager to learn how to compute.

The Reims Champagne Ardenne University offers under-graduate courses dedicated to web-sites design. The courses include design, communication, project management, HTML and CSS languages but also PHP language. The latter is much related to algorithmics and object-oriented programming concepts. Some students want to graduate in these courses without having a computer science background but having a communication background. As a result, some students have difficulties to catch a good mental representation of any algorithm and think it is beyond their skills. Variables, conditional instructions, loops, parameters, functions, objects, polymorphism, every computer science entities will be meaningless words however hard teachers can try to explain. Nick Ellis pointed out in [Ell91] that in the matter of word meaning, "the eyes see vividly, but ears only faintly hear, fingers barely feel and the nose doesn't know". Hopefully, virtual environments can motivate students to engage in learning activities [CWW^{+} 13]. So to help students learn and conceptualize, we created the virtual world of AlgoPath. In this world, variables are 3D figures coming out of huts and carrying backpacks, waiting near bus stops on a road made of bitumen. Statements are roads and functions and procedures are alternative roads - for more details on the concepts of AlgoPath see [PLD12] and [PL13] and for a blink of it see Fig.1.

Figure 1: This is the world of AlgoPath.

The previous versions of AlgoPath could help students avoid common mistakes (such as forgetting to assign a parameter passed by reference or using a variable that is not assigned a value yet in a right value of an assignment) but it couldn’t help them find the statements of an algorithm for a given problem. The truth is there was no intelligence within AlgoPath. In this paper, we present AlgoTch the virtual teacher of AlgoPath built upon the Model-View-Controller architecture (MVC). The role of AlgoTch is to mimic advice, hints and tips a real teacher would give in a ’I-don’t-know-what-to-do’ situation. AlgoTch is computed to help students whether they are discovering AlgoPath for the first time or they are playing against one another to find the best algorithm or they are looking for the best algorithm as a team.

The paper consists of five sections: section 2 overviews the existing models for a virtual teacher and how gameplays affect learning activities. Section 3 explains the inner architecture of AlgoPath. Section 4 describes how we created a virtual teacher into the architecture of AlgoPath an what gameplay it brings. Section 5 explains why we used Unity to implement AlgoTch and section 6 concludes this paper and proposes future work.

2. Overview

In this section, we review how gameplays impact the learning process (section 2.1), how useful a virtual teacher can be when included in a virtual environment (section 2.2) and how it is possible to create a virtual teacher (section 2.3).

2.1. *Impact of gameplays on learning activities*

There are numerous studies dealing with human relations but we focus here on papers analyzing competitive or cooperative aspects of relations between individuals or groups. The first studies that had a concern about that matter were conducted in 1949 (see for example Deutsch [Deu49a] [Deu49b]). Relations initiated in classrooms to induce competitive or cooperative relations obey the same principles that exist among humans in everyday life. [QJJ95] studies relations between students involved in the learning of problem-solving and shows that cooperative learning is a better way to solve non linguistic problems such as mathematics. Exchange of information and recommendations between students flourish ideas and sharing and problems are solved more rapidly than in a thinking retreat. [AH10] shows the usefulness of cooperative learning in the context of scientific courses and states that cooperative students significantly outperform conventional students.

Nowadays these relations are watched within a computer-made environment that is named virtual learning environment or serious game. Tom Weneck, a games designer and critic, pointed out in [Kun12] that games are emblematic of the time they come from. Bacause social media are now embedded in our lives, learning environments and serious (or not) games naturally appeared. They foster relations between players in order to make them succeed only if they cooperate with one another.

Anyway, not all serious games engage students in committed learning because learning activities have to be integrated inside the game story but also outside of it when the computer is off [Ke08]. Game stories are important to keep learners committed but the gameplay is essential too. Gamers engaged in collaborative activities spend significantly more time in the game and, by doing so, get a higher quality result [Ham08]. Similarly, when gamers play with friends they put additional efforts in the game and they are better committed [PH12]. So we can assume that playing with other classmates have an interesting result on learning activities.

Non-linguistic learning takes benefit from collaborative gameplay [Ke08]. Collaborative and competitive gameplays can occurred simultaneously in games [WGS12]. For example, the game "Escape from Wilson Island" involves teams [WGGS12] so the gameplay focuses on collaborative tasks in order to force players coordinations to succeed.

2.2. *A virtual teacher, what good does it do?*

Through entertainment, a serious game or a virtual learning environment is a tool to engage players in learning activities. Learning activities may involve a teacher. Teachers in classrooms can provide feedback to students. Positive cognitive feedback was shown to increase students' perceived self-efficacy, and cognitive feedback in general was shown to promote learning gains [BPW$^+$ 08]. A virtual teacher should do the same and provides information that help students, provides opportunities for students to self-correct and praises effort and focuses on learning goals [SBGB13]. The virtual environment can determine the learning style of an individual and propose the most appropriate content based on her/his errors and/or on her/his progress during the process of learning. For example, [FGLVM13] focuses on the structure of the courses and the way the student learns to adapt the chronology of the ongoing needs of the student. The e-teacher of [SGA08] assists students in the choice of their courses. The content is adapted to their way of learning [OOOBG4S] [XW06]. The intelligent tutoring system described in [FBN12] helps students during their learning progress. It changes the content according to the students' skills and knowledge. In [BQ11], the act of the virtual teacher is performed by an expert system. The latter is decomposed into two parts. The first tracks and identifies the errors made by students. The second is a pedagogical model in order to assist students int their learning activities. How to achieve this role (providing an assistance) is the topic of the next section.

2.3. *Intelligent agents*

Having a teacher by her/his side would be beneficial for the user if he/she would get lost when, for example, he/she would discover AlgoPath for the first time. Such a good Samaritan could be a patient and merciful teacher. A teacher can be virtualized by an intelligent agent. An intelligent agent is an entity in a software or in a part of one supposedly able to act by itself [Sch05]. This entity is situated in an environment, is rather autonomous in the sense that it can react to changes in the environment, and is able to perform reasoning and by doing so to determine actions. Intelligent agents are and have been used in games, simulation games or simulators. See for example the serious game Roma Nova whose goal is to teach history to young individuals and, for that purpose uses pedagogical conversational agents [FL11]. The goal of an

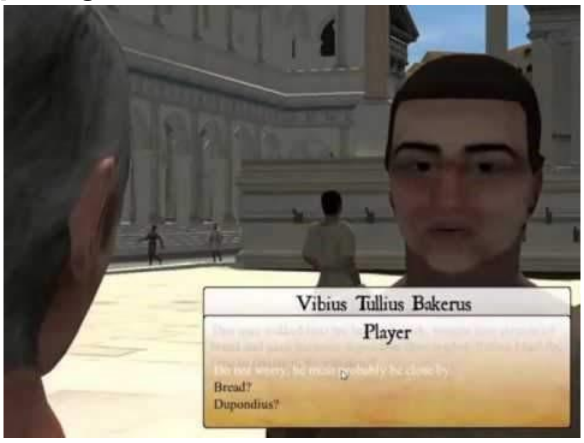


Figure 2: Serious game Roma Nova developed at the Serious Games Institute.

intelligent agent is to balance its external environment and its internal environment in order to make them meet [TP99]. In a learning environment such as AlgoPath it would mean that the external environment would be the user, the internal environment would be the game and the goal of the virtual teacher would be to help the user changes the game in a way the virtual world would represent the algorithm to be created.

When the virtual teacher is represented by an intelligent agent, the latter can answer questions asked by the user. In AlgoPath, the first available question would be: "What can I do?". To answer such a question, predicate logic is used to formalize knowledge [NM14]. Formulas of predicate logic contain variables which can be quantified. They let us express natural language sentences that often contain a conditional statement and a hypothesis. For example, the sentence "Every Bourgueil wine has a red hue" - which is quite true but not entirely because 2% of Bourgueil wine has a less red shade - can be expressed by a formula similar to $8xBOURGUEIL(x)\ )\ HUE(x;RED)$.

The state of the environment is expressed by a set of facts. Knowledge is a set of formulas commonly named rules. Reasoning is achieved when new facts are discovered after having considered every possible rules.

Reasoning can be performed forward starting from a fact in order to discover new ones or backward when a specific goal is to be reached. A fact can also be proved by adding its negation to the knowledge base and by reaching a contradiction. Inference can be done by using several normalized rules:

– the "Modus Ponens" rule $(P\ \hat{}\ (P\ )\quad Q))\ )\quad Q$ that stands for if P is asserted to be true and P implies Q, so therefore Q must be true;

– the "Modus Tollens" rule $((P\ )\ Q)\ \hat{}\ :Q)\ )\ :P$ that stands for if P implies Q and Q is false then P must be false;

– the material implication rule $(P\ )\ Q)\ ,\ (:P\ \_\ Q)$ that states that P implies Q is true is equivalent to the negation of P or Q is true.

3. Inner Model of AlgoPath

In this section, we want to show how the inner model of AlgoPath is structured in order to understand how it is possible (1) to develop AlgoPath in a distributed context and (2) to include AlgoTch in it.

AlgoPath is based on the Model-View-Controller architecture (see Fig.3) [Ree07]. MVC was conceived as a general solution to the problem of users controlling a large and complex data set, which is usually the case with games. MVC consists of three kinds of objects. The Model is the application object, the View is its screen presentation, and the Controller defines the way the user interface reacts to user input. MVC is known to increase flexibility and reuse. A View must

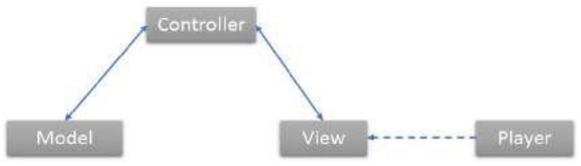
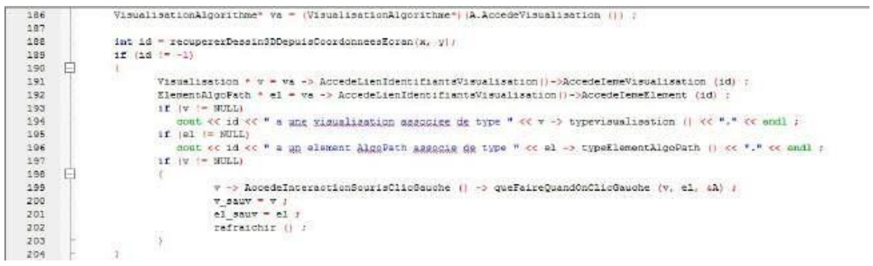


Figure 3: The MVC architecture.

ensure that its appearance reflects the state of the Model. Whenever the Model's data change (generally in response to a user interaction), the Model - through the Controller - notifies Views that depend on it. In response, each View can update itself. MVC is also known to handle pretty well multiple Views to a Model to provide different presentations. In AlgoPath every algorithmic entity (variables, types, statements, ...) has a visual representation. We build, among other hierarchies, one class hierarchy for algorithmic entities whose top class is named "ElementAlgoPath" and one visualization class hierarchy whose top class is unsurprisingly named "Visualization". Thanks to the MVC architecture, algorithmic entities such as assignments for example can have multiple views whose correct one is chosen within the context application (whether the user is in an imperative programming context or in an object-oriented programming context). As the visualization is totally independent from the model, this architecture is also suitable if we may decide to change the world theme.

Thanks to object-oriented programming, the Controller part is very synthetic (see Fig.4). It only has to find which visualization part the user clicked onto, determine the corresponding "ElementAlgoPath" data and launch the appropriate interaction. In a "Visualization" class, several interactions are stored (when the user clicks on the left button, when the user clicks on the right button and so on if necessary). Consequently, we have a third class hierarchy dedicated to interactions.



```
VisualisationAlgorithme* va = (VisualisationAlgorithme*) (A.AccedeVisualisation ()) ;

int id = recupererDessin3DDepuisCoordonneesEcran(x, y) ;
if (id != -1)
{
        Visualisation * v = va -> AccedeLienIdentifiantsVisualisation()->AccedeIemeVisualisation (id) ;
        ElementAlgoPath * el = va -> AccedeLienIdentifiantsVisualisation()->AccedeIemeElement (id) ;
        if (v != NULL)
           cout << id << " a une visualisation associee de type " << v -> typevisualisation () << "." << endl ;
        if (el != NULL)
           cout << id << " a un element AlgoPath associe de type " << el -> typeElementAlgoPath () << "." << endl ;
        if (v != NULL)
        {
                v -> AccedeInteractionSourisClicGauche () -> queFaireQuandOnClicGauche (v, el, &A) ;
                v_sauv = v ;
                el_sauv = el ;
                refraichir () ;
        }
}
```

Figure 4: A portion of the code of the Controller dedicated to left clicks in AlgoPath.

4. AlgoTch

The previous versions of AlgoPath ([PLD12] and [PL13]) were developed for a single player. AlgoPath could be used for personal entertainment or in a classroom. When used for personal entertainment, the user could click here and there but if she/he had not a goal in mind, she/he could be bored pretty quickly. AlgoPath was a world of silence. That is the reason why we wanted to add the feeling that the user was not alone by implementing a virtual teacher helping her/him to take the game in hand. When used in classrooms, a teacher needed to state out

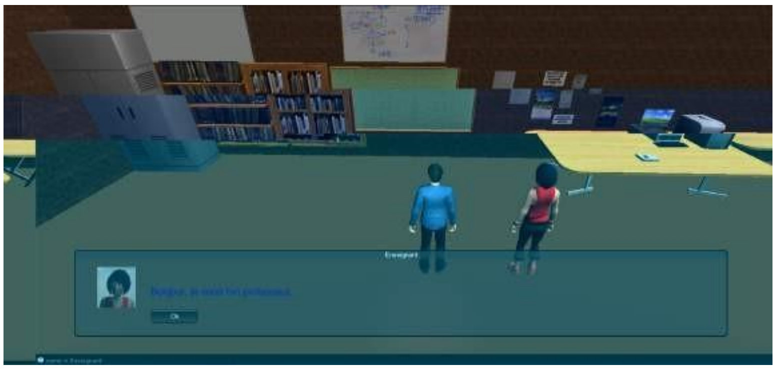
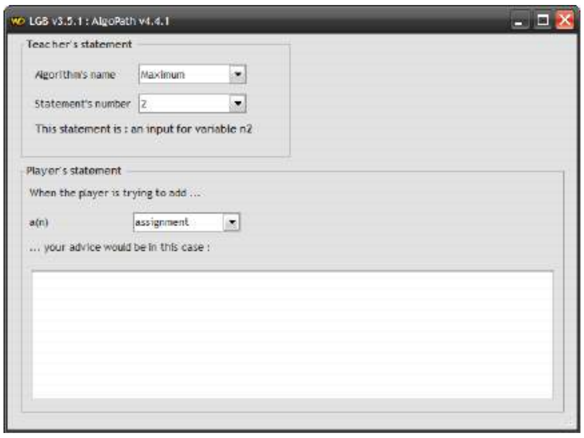


Figure 5: AlgoTch accompanies the avatar of the student within AlgoPath.

– The real teacher can now design her/his algorithms into AlgoPath that are stored in a database added in the model of AlgoPath. This feature allows the real teacher to pick a precise algorithm at a beginning of a session or to let AlgoPath does it randomly for her/him.

– AlgoTch is designed within AlgoPath. AlgoTch is only available if the player chooses to take it with her/him along the game (see Fig.5). AlgoTch can help the student when she/he discovers AlgoPath for the first time or if she/he chooses to design an algorithm AlgoTch chose for her/him. In both cases, AlgoTch supervises every interactions. So it is directly linked with the Controller part of the MVC architecture (see section 3). Therefore AlgoTch is a super controller. It changes the course of events within AlgoPath. It proposes advice and hints in order to help the player find the best interaction. If necessary, at any time, the user can ask AlgoTch questions such as "What can I do now?". Then, knowing the logic of designing algorithms the virtual teacher can propose things to do to the student. For example, it can answer "You can declare a variable. A variable stores values. The declaration of a variable is performed by clicking on huts. There are 5 huts. The one with the blue roof declares a variable that stores an integer, the one with the purple roof declares a variable that stores a real, [...] that stores a string. Go knock on a door!". As answers like this are totally related to logic, the real teacher cannot sway the sayings of AlgoTch. But it is totally different when the user makes a mistake when designing an algorithm chosen by the real teacher. Logic can lead to a logical algorithm. The methodology is correct and the syntax is correct but nobody knows what the algorithm does. Algorithms have a purpose. They are designed for a precise goal. Therefore the logic of any algorithms does not suffice anymore. The goal has to be taken into account. Then when a mistake occurs, it is important to explain to the user why there is a mistake considering the algorithm in progress. That is the reason why we added the feature explained in the next item.


LGB v3.5.1 : AlgoPath v4.4.1

Teacher's statement

Algorithm's name: Maximum

Statement's number: 2

This statement is : an input for variable n2

Player's statement

When the player is trying to add ...

a(n): assignment

... your advice would be in this case :

Figure 6: The interface to set the virtual teacher for an algorithm.

– Each teacher has her/his own way of teaching. We do not want AlgoTch to be a barrier from a way of teaching. So AlgoPath proposes a tool in which the teacher can express her/his own words (see Fig. 6). After designing her/his own algorithms, the teacher has to fill a database of advice if the player is wrong. AlgoTch reviews each interaction made by the teacher when he designed the algorithm, leading to an ElementAlgoPath. It calls to mind each one of them. AlgoTch asks her/him: "What would you say when the player is wrong making this precise interaction?". Each time, there are only six possible wrong interactions. The answer is either a tip leading to the right interaction or a specific comment regarding the wrong interaction. Let’s take an example with first two statements being inputs and students interacting with the wrong statements boxes. Either the real teacher writes the same advice for all the wrong interactions (for example, something like "Think of asking the user the values first") or she/he writes a specific comment ("It is good to think of setting variables but ask the user instead" if the player interacts with the assignment box instead of the input box). For this feature, the process of finding an algorithm must be seen in a linear way even if designing an algorithm is not necessarily sequential and continuous. We do not want to take into account every possible order leading to the right algorithm - at first anyway. But this assumption is moderated for the creations of variables. Variables of an algorithm belong to three sets: the input, the output and the others. Before expressing the statements of an algorithm, students should answer two questions: (1) what are the input variables? (2) what are the output variables? Once students have answered these questions, they can create the variables in any order they want. During the design of an algorithm, students can also create variables at any time. In the end, AlgoTch only checks whether students have created the right number and the right type of input variables, output variables and others. We are aware that the real teacher has some preparatory work to do but once the game is on, AlgoTch acts on its own. The way teachers teach is very personal and we thought that letting them express their own words is the best option.

– The evaluation of students is performed by calculating a score according to the number of errors students made - obviously, the fewer they made, the better the score is.

4.2. *Model*

In the next two sections, we explain how we modified the general architecture of AlgoPath and what we added to it to fully implement the features described previously (see section 4.1). Two events must be considered:

– The first event is when AlgoTch realizes a student has made a mistake. She/he has just interacted but it is not the right interaction. The hints and tips given by AlgoTch are directly linked to the algorithm in progress and the pedagogy of the real teacher.

– The second event is when a student asks a question. The student does not know what to do and asks prior to any interaction. The hints and tips given by AlgoTch are linked to logic.

4.2.1. *AlgoTch realizes an interaction is not right*

The real teacher creates algorithms and states sentences for AlgoTch. The student tries to find the algorithm the teacher or AlgoPath selected. The Controller of the MVC architecture catches interactions. If necessary, it passes them to AlgoTch. AlgoTch accesses the real teacher's sentences. AlgoTch detects if the interaction is correct or not. If it is, the whole system acts as if AlgoTch were not there. If it is not, AlgoTch stops the process, disrupts the system and takes over the actions of the View. The View displays the specific guidance in correlation with the stopped interaction. Another interaction may occur again and the system iterates. See Fig.7 for an overview of this process. Fig.7 also

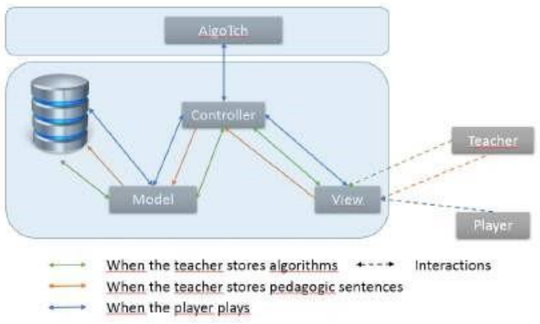

AlgoTch
Controller
Teacher
Model
View
Player
When the teacher stores algorithms
Interactions
When the teacher stores pedagogic sentences
When the player plays

Figure 7: Integration of AlgoTch inside the MVC architecture.

summarizes three modes:

– When the teacher stores algorithms to be further used in the classroom or at home. In that case, AlgoTch is off.

– When the player games. In that case, AlgoTch is on.

– When the teacher stores pedagogic sentences she/he would say in the classroom if errors were to be made by students. In that case, AlgoTch is off.

4.2.2. *A student asks a question*

Logic of any algorithm is stored within AlgoTch using first order logic (read section 2.3 to see how it can be done theoretically). What follows is the set of rules we created to formalize this logic. We only present rules related to declarations of variables, assignments, conditional statements and loops (obviously it goes on with functions and procedures and structures but rules relative to these concepts are not mentioned here):

– Rule number 1 is:) *DECLARAT IONS_ARE_ALLOW ED*. It means that when nothing has been done yet, students can declare variables.

– Rule number 2 is: 8v : IS_DECLARED(v)) CAN_BE_DECLARED(v). It means that if a variable named *v* is not declared yet then it is possible to declare a variable named *v*.

– Rule number 3 is: 8v IS_DECLARED(v)) CAN_BE_ASSIGNED(v). It means that if a variable named *v* is declared then it is possible to assign a value to *v*.

– Rule number 4 is: 8v IS_DECLARED(v)) CAN_BE_INPUT(v). It means that if a variable named *v* is declared then it is possible to input *v*.

– Rule number 5 is: 9v IS_ASSIGNED(v)) (OUTPUTS_ARE_ALLOW ED ^ CONDIT IONALS_ARE_ALLOW ED ^ LOOP S_ARE_ALLOW ED). It means that if there exists a variable that is assigned a value then the user can create outputs, conditional statements and loops. Obviously, at first anyway, only statements on *v* will be allowed. This is controlled by rules numbered 7, 10 and 11.

– Rule number 6 is: 9v IS_DECLARED(v)) (ASSIGNMENTS_ARE_ALLOW ED ^ INPUTS_ARE_ALLOW ED). It means that if somehow the user succeeded in creating the declaration of a variable, then assignments and inputs are now possible. Obviously, at first anyway, only statements on *v* will be allowed. This is controlled by rules numbered 3 and 4.

– Rule number 7 is: 8v IS_ASSIGNED(v)) CAN_BE_OUPUT(v). It means that if the variable *v* is assigned a value then students can output it.

– Rule number 8 is: 8v 9x (CAN_BE_ASSIGNED(v) ^ IS_ASSIGNED(x))) CAN_BE_USED_IN_RV ALUE(v;x). It means that if the variable named *x* is assigned and if an assignment on variable named *v* is to be performed then the variable *x* can be used in the right value of the assignment.

– Rule number 9 is: 9v (IS_DECLARED(v) ^ INT EGER(v) ^ LOOP S_ARE_ALLOW ED)) COUNT_LOOP S(v). It means that if there exists a variable named *v* that is declared and is an integer then the user can create count loops whose exit condition is based on *v*.

– Rule number 10 is: 9v IS_ASSIGNED(v)) CONDIT IONAL_LOOP S(v). It means that if there exists a variable named *v* that is assigned then the user can create conditional loops whose exit condition is based on *v*.

– Rule number 11 is: $\exists v \; IS\_ASSIGNED(v) \Rightarrow CONDITIONAL\_STATEMENTS(v)$. It means that if there exists a variable named v that is assigned a value then the user can create conditional statements whose conditions are based on v.

Within this new version of AlgoPath, students can only ask one question that is "What can I do?". When this question is asked, AlgoTch launches the inference engine. Moreover, every time the user interacts, the corresponding fact is added to the knowledge base. The inference engine acts as follows:

1) It detects all the rules that match the knowledge base.
2) It selects everyone of them and executes them. The order it applies them is the order of the numbers of the rules. It adds new facts in the knowledge base.
3) It iterates until no new rules are a match.
4) It gives new facts to AlgoTch.
5) The virtual teacher expresses the answers to the user.

Tables 1, 2 and 3 show the answers AlgoTch can say to the user when a new fact is discovered.

Let us see how it goes with a simple example that relies on the algorithm shown in Fig.8. Let us suppose that the user can create by her/his own this algorithm but right after that she/he stops and does not know what to do anymore.

At the beginning of the algorithm, the knowledge base is empty but when the user begins to move the avatar, a first step of inference begins (see Fig.9.a).

It brings the fact $DECLARATIONS\_ARE\_ALLOWED$ into the knowledge base (see Fig.9.b). The first interaction the user makes is the declaration of the variable $n1$. The fact $IS\_DECLARED(n1)$ is added to the knowledge base (see Fig.9.c) and the inference begins.

New Fact	AlgoTch's Sentence

DECLARAT IONS_ARE_ALLOW ED	You can declare variables. They store values. The declaration of a variable is performed by clicking on 5 huts. The one with the blue roof declares a variable that stores an integer, the one [...] stores a string. Go knock on some doors!
ASSIGNMENTS_ARE_ALLOW ED	You can assign values to variables. Values must be of the same type as variables. Only real variables that can store integers or reals. Assignments are created by clicking on the box (image here). Go click on that box!
INPUTS_ARE_ALLOW ED	When you do not know what values you can assign to variables, you can ask the future user of your algorithm. Such an action is called an input. Inputs are created by clicking on the box (image here). Go click on that box!
OUTPUTS_ARE_ALLOW ED	Now, you can show values to the future user of your algorithm. Showing her/him values is called outputs. Outputs are created by clicking on the box (image here). Go click on that box!

Table 1: Sentences said by AlgoTch when a new fact is discovered.

New Fact	AlgoTch's Sentence

COND:_ARE_ALLOW ED	Great, you can create conditional statements from now on. They allow your algorithm to execute statements based on a decision. They are created by clicking on the box (image here). Go click on that box!
LOOP S_ARE_ALLOW ED	You can also create loops. They repeat continually statements until a condition is finally reached. They are created by clicking on the box (image here). Go click on that box!
CAN_BE_DECLARED(v)	Fact not available since the only question is "What can I do?".
CAN_BE_ASSIGNED(v)	If you don't know what assignment to create, you can assign a value to the variable named v.
CAN_BE_INPUT(v)	If you don't know what input to create, you can ask the future user of your algorithm the value he wants to assign to the variable v.

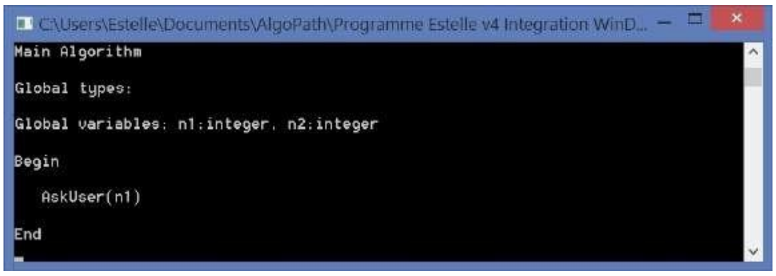
Table 2: Sentences said by AlgoTch when a new fact is discovered.

New Fact	AlgoTch's Sentence
CAN_BE_OUTPUT(v)	If you don't know what output to create, you can show the future user of your algorithm the value stored in the variable v.

CAN_BE_USED_IN_RV ALUE(v;x)	You can use the variable named x to assign a value to the variable named v.
COND:_STATEMENTS(v)	You can create a conditional statement whose condition involves the variable named v.
COUNT_LOOP S(v)	You can create a count loop whose counter is the variable named v.
CONDIT IONAL_LOOP S(v)	You can create a conditional loop whose condition involves the variable named v.

Table 3: Sentences said by AlgoTch when a new fact is discovered.

Rules numbered 1, 3, 4 and 6 are a match and new facts are added to the knowledge base (see Fig.9.d). Because the user does not ask a question, AlgoTch stays quiet. Thanks to the second interaction, the user declares variable $n2$. The fact $IS\_DECLARED(n2)$ is added to the knowledge base (see Fig.9.e). The inference processes again to get new facts to the knowledge base ((see Fig.9.f). Then finally, the student creates the statement to input the variable $n1$. The fact $IS\_ASSIGNED(n1)$ is added to the knowledge base ((see Fig.9.g). The inference detects that rules numbered 5, 7, 8, 9, 10 and 11 are a match and finally gives the knowledge base in Fig.9.h. And then the user asks "What can I do now?". Then there are two cases. Either the entire process is delivered to the student or only sentences related to the last new facts are said to the student. The choice belongs to her/him. AlgoTch's sentences that will be said to the student if she/he chooses to only have advice from her/his last action can be shown in Fig.10.


C:\Users\Estelle\Documents\AlgoPath\Programme Estelle v4 Integration WinD...

```
Main Algorithm

Global types:

Global variables: n1:integer, n2:integer

Begin

   AskUser(n1)

End
```

Figure 8: The algorithm of the example on which rules are inferred.

$$a)\ KB = \{\emptyset\}$$

$$b)\ KB = \{DECLARATIONS\_ARE\_ALLOWED\}$$

$$c)\ KB = \left\{\begin{matrix} DECLARATIONS\_ARE\_ALLOWED, \\ IS\_DECLARED(n1) \end{matrix}\right\}$$

$$d)\ KB = \left\{\begin{matrix} DECLARATIONS\_ARE\_ALLOWED, \\ IS\_DECLARED(n1), \\ CAN\_BE\_ASSIGNED(n1), \\ CAN\_BE\_INPUT(n1), \\ ASSIGNMENTS\_ARE\_ALLOWED, \\ INPUTS\_ARE\_ALLOWED \end{matrix}\right\}$$

$$e)\ KB = \left\{\begin{matrix} DECLARATIONS\_ARE\_ALLOWED, \\ IS\_DECLARED(n1), \\ CAN\_BE\_ASSIGNED(n1), \\ CAN\_BE\_INPUT(n1), \\ ASSIGNMENTS\_ARE\_ALLOWED, \\ INPUTS\_ARE\_ALLOWED, \\ IS\_DECLARED(n2) \end{matrix}\right\}$$

$$f)\ KB = \left\{\begin{matrix} DECLARATIONS\_ARE\_ALLOWED, \\ IS\_DECLARED(n1), \\ CAN\_BE\_ASSIGNED(n1), \\ CAN\_BE\_INPUT(n1), \\ ASSIGNMENTS\_ARE\_ALLOWED, \\ INPUTS\_ARE\_ALLOWED, \\ IS\_DECLARED(n2), \\ CAN\_BE\_ASSIGNED(n2), \\ CAN\_BE\_INPUT(n2) \end{matrix}\right\}$$

$$g)\ KB = \left\{\begin{matrix} DECLARATIONS\_ARE\_ALLOWED, \\ IS\_DECLARED(n1), \\ CAN\_BE\_ASSIGNED(n1), \\ CAN\_BE\_INPUT(n1), \\ ASSIGNMENTS\_ARE\_ALLOWED, \\ INPUTS\_ARE\_ALLOWED, \\ IS\_DECLARED(n2), \\ CAN\_BE\_ASSIGNED(n2), \\ CAN\_BE\_INPUT(n2), \\ IS\_ASSIGNED(n1) \end{matrix}\right\}$$

$$h)\ KB = \left\{\begin{matrix} DECLARATIONS\_ARE\_ALLOWED, \\ IS\_DECLARED(n1), \\ CAN\_BE\_ASSIGNED(n1), \\ CAN\_BE\_INPUT(n1), \\ ASSIGNMENTS\_ARE\_ALLOWED, \\ INPUTS\_ARE\_ALLOWED, \\ IS\_DECLARED(n2), \\ CAN\_BE\_ASSIGNED(n2), \\ CAN\_BE\_INPUT(n2), \\ IS\_ASSIGNED(n1), \\ OUTPUTS\_ARE\_ALLOWED, \\ CONDITIONALS\_ARE\_ALLOWED, \\ LOOPS\_ARE\_ALLOWED, \\ CAN\_BE\_OUPUT(n1), \\ CAN\_BE\_USED\_IN\_RVALUE(n2, n1), \\ COUNT\_LOOPS(n1), \\ COUNT\_LOOPS(n2), \\ CONDITIONAL\_LOOPS(n1), \\ CONDITIONAL\_STATEMENTS(n1) \end{matrix}\right\}$$

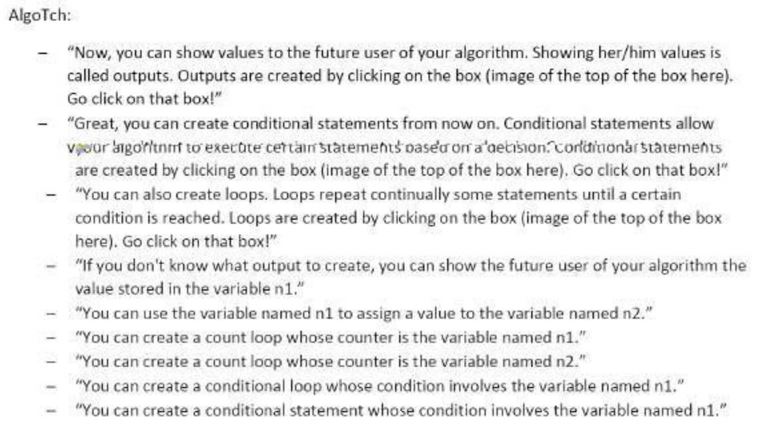
Figure 9: Evolution of the knowledge base.

4.3. *Gameplay*

As shown in [SZ99], a virtual environment is a 2D or a 3D virtual world (real or not). Virtual environments incorporate realistic graphics to create an immersive experience. Networked virtual environments are computer systems that create virtual worlds, where users can interact in real time both with the game and with the other users connected. Users are connected to the Internet and work on different computers, accessing the same virtual scene. The architectures that support these types of environments usually fall into one of the following categories:

1) client-server architectures, where the client communicates its changes to the server and the server, in turn, communicates the changes to all the other clients;

2) or, peer-to-peer architecture where the clients communicate directly with one another [MD03].



AlgoTch:

- "Now, you can show values to the future user of your algorithm. Showing her/him values is called outputs. Outputs are created by clicking on the box (image of the top of the box here). Go click on that box!"
- "Great, you can create conditional statements from now on. Conditional statements allow your algorithm to execute certain statements based on a decision. Conditional statements are created by clicking on the box (image of the top of the box here). Go click on that box!"
- "You can also create loops. Loops repeat continually some statements until a certain condition is reached. Loops are created by clicking on the box (image of the top of the box here). Go click on that box!"
- "If you don't know what output to create, you can show the future user of your algorithm the value stored in the variable n1."
- "You can use the variable named n1 to assign a value to the variable named n2."
- "You can create a count loop whose counter is the variable named n1."
- "You can create a count loop whose counter is the variable named n2."
- "You can create a conditional loop whose condition involves the variable named n1."
- "You can create a conditional statement whose condition involves the variable named n1."

Figure 10: AlgoTch's advice in case of Fig.8 algorithm.

Papers such as [TZ07][ZCLT04] address the problems of network bandwidth, CPU cycles, damage of the interactivity and consistency of the game, ..., for such environments.

Having such an architecture, AlgoPath can be used by several players or not. These players can be with a real teacher or not. Then the following situations can occur:

1) The player is self-thinking at home. She/he can discover AlgoPath by her/his own or if AlgoPath has been introduced in school she/he can try again afterschool.

2) The player is self-thinking in a classroom. She/he can try an algorithm of her/his own if she/he is allowed to do that or the teacher has prepared her/his exercise within AlgoPath and she/he has to find it.

3) The player is in a collaborative gameplay. An algorithm is not necessarily chosen. Players design the same algorithm altogether. Consequently, Views are the same on every computer. AlgoTch scans the interactions of every player. The first correct interaction that it receives updates the Views of every player. The player who gives the correct answer scores. Anyhow, the players receive personal advice from AlgoTch if their interactions are not correct. The game ends when the algorithm is complete.

4) The player is in a competitive gameplay. As in the collaborative gameplay, an algorithm is necessarily chosen but the winner is the one who finds the whole correct algorithm first. In that case, Views are independent and advice personal. The game is finished when there is a winner. Every player scores according to the rate of correct interactions she/he did during the play.

In these fourth cases, instead of taking a pen and a paper, each student begins to play with AlgoPath and AlgoTch helps them go through the process at any time if necessary.

5. Implementation

The first versions of AlgoPath were standalone applications. The 3D content was featured thanks to the VTK library [vtk]. VTK consists of a C++ class library for 3D computer graphics, image processing and visualization. AlgoPath is now a distributed application therefore players can access it at home or during courses in classrooms. Because students are generally not equipped with the same kind of hardware (computers, tablets or smartphones), putting AlgoPath on the web seems relevant.

In recent years, 3D graphics have become an increasingly important part of the multimedia web experience. A large number of technologies for the development of 3D content can be found on the web [ERB+ 14]:

– VRML, X3D and ISO standards: VRML was developed in 1994 and replaced by X3D in 2004. Several applications and browser plugins were developed to enable the display of VRML scenes in the browser (CosMo Player, WorldView, VRMLView, ...). X3D was designed to be compatible with VRML.

– X3Dom was introduced in 2009. It is designed to work with web standards such as Ajax.

– XML3D is very similar to X3Dom. It proposes to extend the existing properties of HTML wherever possible, embedding 3D content into a web page, with the maximum reuse of existing features.

– Xflow is a solution to process intensive data on the web. It is integrated into XML3D. It provides character animation, simulations of various types (fire, smoke, air movements, ...).

– CSS 3D enables to format elements using 3D transforms (rotations, scales, translations). But CSS 3D cannot perform true light and shadow.

– Google O3D is an open-source JavaScript API for creating interactive 3D applications in the browser. It is viewed as bridging the gap between desktop based 3D accelerated graphics applications and HTML based web browsers.

– WebGL (Web Graphics Library) is a JavaScript API for rendering interactive 3D graphics within any compatible web browser without the use of plug-ins. It is intended to be used by experienced graphics programmers. Several libraries originated from WebGL such as SpideGL or LightGL or Three.js.

– Java3D features a full scene graph and is able to render using Direct3D or OpenGL. JOGL (Java OpenGL) and LWJGL (LightWeight Java Game Library - it featured Minecraft, the popular multiplatform game) are libraries that provide 3D graphics, 3D sound and controllers (joysticks, ...) to applications.

The selection of the appropriate technology depends on the needs and requirements of the application developed. Unity [uni] is a proprietary videogame engine which, compared to the technologies listed above, is highly superior, aiming to be an application to create video games. It is powerful and fully integrated with a complete set of tools and rapid workflows to create interactive 3D and 2D contents. So we chose Unity to develop AlgoPath in a distributed way. The implementation is at its beginning. For us, one of the main interests of Unity is that a Unity scene can be created entirely in code. Unity accepts C# or JavaScript code but it is also possible to import our own library. The latter lets us import the M-part and the C-part of the MVC model of AlgoPath, the V-part being totally handled by Unity.

6. Conclusion and future work

In this paper, we explained how we made the model of AlgoPath evolved in order to include AlgoTch. AlgoPath is now a distributed virtual environment in which students receive advice whether they are playing alone or in a collaborative or competitive way. AlgoTch intervenes whenever necessary to provide players with hints and tips.

What AlgoTch says comes either from a real teacher who has to specify her/his own words within AlgoPath or from algorithmics logic. This new version of AlgoPath is still in progress and is currrently implementing using Unity, a videogame engine.

Future work will focus on realizing focus groups so that AlgoPath can be properly evaluated. Some have already begun with the University of Lorraine in France. Results will show if the gameplay seems good for teachers and students or not. Moreover, we want to improve the quality of AlgoTch. We stated that AlgoTch has to follow the flow of statements and therefore, so have the players when in practice programmers do not think straight but go back and forth. We limited the number of questions the user could ask to one ("What can I do now?"). Obviously more elaborate questions will have to be implemented. More thoughts will also have to be given to implement oriented-object programming.

R eferences

[AH10] Zahara Aziz and Md. Anowar Hossain. A comparison of cooperative learning and conventional teaching on students’ achievement in secondary mathematics. *Procedia - Social and Behavioral Sciences*, 9(0):53 – 62, 2010. World Conference on Learning, Teaching and Administration Papers.

[BPW+ 08] Kristy Elizabeth Boyer, Robert Phillips, Michael Wallis, Mladen Vouk, and James Lester. Balancing cognitive and motivational scaffolding in tutorial dialogue. In Beverley P. Woolf, Esma Aimeur, Roger Nkambou, and Susanne Lajoie, editors, *Intelligent Tutoring Systems*, number 5091 in Lecture Notes in Computer Science, pages 239–249. Springer Berlin Heidelberg, January 2008.

[BQ11] Cedric Buche and Ronan Querrec. An expert system manipulating knowledge to help human learners into virtual environment. *Expert Systems with Applications*, 38(7):8446 – 8457, 2011.

[CWW+ 13] Michael Chau, Ada Wong, Minhong Wang, Songnia Lai, Kristal W.Y. Chan, Tim M.H. Li, Debbie Chu, Ian K.W. Chan, and Wai ki Sung. Using 3d virtual environments to facilitate students in constructivist learning. *Decision Support Systems*, 56(0):115 – 121, 2013.

[Deu49a] Morton Deutsch. A Theory of Co-operation and Competition. *Human Relations*, 2(2):129–152, April 1949.

[Deu49b] Morton Deutsch. An experimental study of the effects of co-operation and competition upon group process. *Human relations*, 1949.

[Ell91] Nick Ellis. Chapter 21. word meaning and the links between the verbal system and modalities of perception and imagery or in verbal memory the eyes see vividly, but ears only faintly hear, fingers barely feel and the nose doesn't know. In Robert H. Logie and Michel Denis, editors, *Mental Images in Human Cognition*, volume 80 of *Advances in Psychology*, pages 313 – 329. NorthHolland, January 1991.

[ERB+ 14] Alun Evans, Marco Romeo, Arash Bahrehmand, Javi Agenjo, and Josep Blat. 3d graphics on the web: A survey. *Computers & Graphics*, 41(0):43 – 61, 2014.

[FBN12] Fernando A. Mikic Fonte, Juan C. Burguillo, and Martin Llamas Nistal. An intelligent tutoring module controlled by {BDI} agents for an e-learning platform. *Expert Systems with Applications*, 39(8):7546 – 7554, 2012.

[FGLV M13] Beatriz Fernandez-Gallego, Manuel Lama, Juan C. Vidal, and Manuel Mucientes. Learning analytics framework for educational virtual worlds. *Procedia Computer Science*, 25(0):443 – 447, 2013. 2013 International Conference on Virtual and Augmented Reality in Education.

[FL 11] Sara de Freitas and Fotis Liarokapis. Serious games: A new paradigm for education? In Minhua Ma, Andreas Oikonomou, and Lakhmi C. Jain, editors, *Serious Games and Edutainment Applications*, pages 9–23.
Springer London, January 2011.

[Ham08] Raija Hamalainen. Designing and evaluating collaboration in a virtual game environment for vocational learning. *Computers & Education*, 50(1):98 – 109, 2008.

[K e08] Fengfeng Ke. A case study of computer gaming for math: Engaged learning from gameplay? *Computers & Education*, 51(4):1609 – 1620, 2008.

[K un12] V. K. Kuntz. Bretter, die die welt bedeuten, November 2012.

[MD03] Zyda M. Brutzman D. McGregor D., Kapolka A. Requirements for large-scale networked virtual environments. *Telecommunications, 2003. ConTEL 2003. Proceedings of the 7th International Conference on*, 1:353– 358, June 2003.

[NM14] Faria Nassiri-Mofakham. How does an intelligent agent infer and translate? *Computers in Human Behavior*, 38:196–200, September 2014.

[OOOBG4S] Õzcan Õzyurt, Hacer Ozyurt, Adnan Baki, and Bulent

Guven. Integration into mathematics classrooms of an adaptive and intelligent individualized e-learning environment: Implementation and evaluation of {UZWEBMAT} . *Computers in Human Behavior*, 29(3):726 – 738, 2014S.

[PH12] Wei Peng and Gary Hsieh. The influence of competition, cooperation, and player relationship in a motor performance centered computer game. *Computers in Human Behavior*, 28(6):2100 – 2106, 2012.

[PL 13] Estelle Perrin and Sébastien Linck. Algopath's new interface helps you find your way through common algorithmic mistakes. In *The Sixth International Conference on Advances in Computer-Human Interactions*, pages 188– 193, Nice, France, February 2013.

[PL D12] Estelle Perrin, Sébastien Linck, and Frédéric Danesi. Algopath: A new way of learning algorithmic. In *The Fifth International Conference on Advances in ComputerHuman Interactions*, pages 291–296, Valencia, Spain, January 2012.

[Pre01] Marc Prensky. Digital Natives, Digital Immigrants. *On the Horizon*, 9(5), October 2001.

[QJJ95] Zhining Qin, David W Johnson, and Roger T Johnson. Cooperative versus competitive efforts and problem solving. *Review of educational Research*, 65(2):129–143, 1995.

[Ree07] Trygve Mikjel H Reenskaug. The original MVC reports, February 2007.

[SBGB13] Amy Shannon, Acey Boyce, Chitra Gadwal, and Tiffany Barnes. Effective practices in game tutorial systems. In *Proceedings of the 8th International Conference on the Foundations of Digital Games*, pages 338–345, 2013.

[Sch05] Ralf Schleiffer. An intelligent agent model. *European Journal of Operational Research*, 166(3):666–693, November 2005.

[SGA08] Silvia Schiaffino, Patricio Garcia, and Analia Amandi. eteacher: Providing personalized assistance to e-learning students. *Computers & Education*, 51(4):1744 – 1754, 2008.

[SZ99] Sandeep Singhal and Michael Zyda. *Networked Virtual Environments: Design and Implementation*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1999.

[TP99] Vagan Y. Terziyan and Seppo Puuronen. Knowledge acquisition based on semantic balance of internal and external knowledge. In Ibrahim Imam, Yves Kodratoff, Ayman El-Dessouki, and Moonis Ali, editors, *Multiple Approaches to Intelligent Systems*, number 1611 in Lecture Notes in Computer Science, pages 353–361. Springer Berlin Heidelberg, January 1999.

[TZ07] Duong Nguyen Binh Ta and Suiping Zhou. A twophase approach to interactivity enhancement for largescale distributed virtual environments. *Computer Networks*, 51(14):4131 – 4152, 2007.

[uni] http://unity3d.com.

[vtk] http://www.vtk.org.

[WGGS12] Viktor Wendel, Michael Gutjahr, Stefan Göbel, and Ralf Steinmetz. Designing collaborative multiplayer serious games for collaborative learning. *Proceedings of the CSEDU*, 2012, 2012.

[WGS12] Viktor Wendel, Stefan Göbel, and Ralf Steinmetz. Collaborative learning in multiplayer serious games. In Winfred Kaminski, editor, *Clash of Realities Proceedings 2012*, München, Sep 2012. KoPäd Verlag.

[X W06] Dongming Xu and Huaiqing Wang. Intelligent agent supported personalization for virtual learning environments.
Decision Support Systems, 42(2):825 – 843, 2006.

[ZCLT04] Suiping Zhou, Wentong Cai, Bu-Sung Lee, and Stephen J. Turner. Time-space consistency in large-scale distributed virtual environments. *ACM Trans. Model. Comput. Simul.*, 14(1):31–47, January 2004.